DOCUMENT RESUME

ED 101 690 IR 001 540

| | |
|---|---|
| AUTHOR | Roman, Richard Allan |
| TITLE | Teaching Problem Solving and Mathematics By Computer: An Interim Report. |
| INSTITUTION | Pittsburgh Univ., Pa. Learning Research and Development Center. |
| SPONS AGENCY | National Inst. of Education (DHEW), Washington, D.C.; National Science Foundation, Washington, D.C. |
| REPORT NO | PU-LRDC-1974-15 |
| PUB DATE | 74 |
| NOTE | 69p. |
| EDRS PRICE | MF-$0.76 HC-$3.32 PLUS POSTAGE |
| DESCRIPTORS | *Computer Assisted Instruction; Computer Programs; Educational Research; Elementary Education; Elementary School Mathematics; *Individualized Instruction; *Mathematics Curriculum; Mathematics Instruction; Mathematics Materials; *Performance Based Education; *Problem Solving; Program Descriptions; Teaching Techniques |
| IDENTIFIERS | *MATH FUNCTIONS Program |

ABSTRACT

An interim report from the National Science Foundation describes the FUNCTIONS program--an ongoing effort to teach problem solving and mathematics by computer. Two problems are discussed: How can math content be taught in a manner which also develops problem solving skills? Also, how does a curriculum organized to develop problem solving skills teach math content? The report concludes that both can be taught if math content is structured as a sequence of problems in which students induce organizational rules from examples. The report includes a description of the procedure, the computer programs, and the preparation of math content. According to this report, 88 percent of students taught this way achieve course objectives. (SK)

ED101690

BEST COPY AVAILABLE

# TEACHING PROBLEM SOLVING AND MATHEMATICS BY COMPUTER: AN INTERIM REPORT

Richard Allan Roman

Learning Research and Development Center
University of Pittsburgh

1974/15

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

The research reported herein was supported by a grant from the National Science Foundation (NSF-GJ-510X) and by the Learning Research and Development Center, supported in part as a research and development center by funds from the National Institute of Education (NIE), United States Department of Health, Education, and Welfare. The opinions expressed in this publication do not necessarily reflect the position or policy of the sponsoring agencies and no official endorsement should be inferred. The MATH FUNCTIONS program is the product of many minds working together. David Zarembka analyzed most of the objectives and created the teaching sequences, as well as provided assistance with actual programming. Joan Heller supervised the formative evaluation work in the school and provided incisive comments on student errors. Richard Thackray did statistical work on the data. Nicholas Laudato refined many teaching sequences and helped clarify the methods for making puzzles more complex.

IR 001 540

BEST COPY AVAILABLE

TABLE OF CONTENTS


| | | Page |
|---|---|---|
| I. | Introduction | 1 |
| II. | Procedure | 2 |
| | A. Overview of Procedure | 2 |
| | B. The FUNCTIONS Program | 4 |
| | 1. The Interactions with the Student | 4 |
| | 2. The Kind of Problem Presented | 11 |
| | 3. Problem Solving and FUNCTIONS | 17 |
| | C. Preparation of Mathematical Content | 24 |
| | 1. Functions in Elementary School Mathematics | 24 |
| | 2. The Individualized Mathematics System | 26 |
| | 3. Creating Teaching Sequences | 28 |
| | 4. An Example of a Teaching Sequence | 37 |
| | D. MATH FUNCTIONS and the School | 48 |
| | 1. The Small Computer Resource | 49 |
| III. | The Results | 53 |
| | A. Outcome Statistics | 53 |
| | B. Usage Statistics | 57 |
| IV. | Summary and Implications | 59 |
| | A. Summary | 59 |
| | B. Implications | 60 |
| References | | 63 |

BEST COPY AVAILABLE

## LIST OF TABLES


| Table | | Page |
|---|---|---|
| 1. | Examples of Simple and Complex Rules | 14 |
| 2. | Examples of FUNCTION Problems | 15 |
| 3. | Outcomes on Objectives | 54 |

BEST COPY AVAILABLE

## LIST OF FIGURES


| Figure | | Page |
|---|---|---|
| 1. | The sample display | 4 |
| 2. | The input mode display | 5 |
| 3. | A correct input display | 5 |
| 4. | An incorrect input display | 6 |
| 5. | The test mode display | 7 |
| 6. | A correct test display | 7 |
| 7. | An incorrect test display | 8 |
| 8. | An incorrect change display | 9 |
| 9. | The too-many-input display | 10 |
| 10. | The too-many-tests display | 10 |
| 11. | The tabular form of the display | 19 |
| 12. | Display illustrating subdividing problems | 23 |
| 13. | Sample display for criterion puzzle | 30 |
| 14. | Criterion items for adding fractions and comparing the sum to one | 38 |
| 15. | Instructional puzzles for adding fractions and comparing the sum to one | 41 |
| 16. | Frequency of students seeing each number of objectives in a year | 58 |

BEST COPY AVAILABLE

# TEACHING PROBLEM SOLVING AND MATHEMATICS BY COMPUTER: AN INTERIM REPORT

Richard Allan Roman

Learning Research and Development Center
University of Pittsburgh

## I. Introduction

The MATH FUNCTIONS curriculum package was designed to teach both mathematical content and problem solving skills. The package is one component of an individualized mathematics system operating in an elementary school. All instruction in the curriculum is accomplished through a computer program implemented on the school's computer resource. Other components of the mathematics system teach the same content material, giving students a choice of instructional media. The problem solving curriculum package teaches more than one hundred behavioral objectives, or one-fourth of the elementary school mathematics for grades three through six.

Successfully solving problems should develop inquiry skills. The curriculum package provides a structured environment which encourages problem solving activities such as stating the problem clearly, gathering and organizing data, using feedback, formulating and testing hypotheses, knowing you have finished, dividing problems into subproblems, and combining subsolutions into complete solutions. Students who experience success at solving problems in this structured environment increase their sense of competence and their desire to solve more problems.

This report focuses on two questions. First, how can mathematics content be taught in a manner which also develops problem solving skills? Second, does a curriculum organized to develop problem solving skills

BEST COPY AVAILABLE

teach the mathematical content. A subsequent report will describe how well problem solving skills are taught in the context of mathematical content.

From our research to date, some conclusions can be made. Both mathematical content and problem solving skills can be taught by structuring the mathematical content as a sequence of problems in which the student induces the organizational rule from examples. Appropriate structuring of the environment in which rules are induced teaches problem solving skills. Mathematical content which requires simple rules, simple stimulus material, and for which the student has mastered the prerequisite skills is taught effectively in conjunction with problem solving skills. Many students learn mathematical content in an environment which also teaches problem solving skills.

## II. Procedure

### A. Overview of Procedure

The procedure section contains three parts corresponding to the three major tasks in preparing the curriculum package and introducing it to the school. The first part describes how the computer program (FUNCTIONS) structures the student's environment to teach problem solving. The second part describes how mathematical content material is prepared for presentation via the computer program. The third part describes how the program to teach mathematical content (MATH FUNCTIONS) was used in a public school. The following paragraphs give an overview of the three parts. The rest of this section is divided into three additional subsections that deal intensively with each of the three parts.

The FUNCTIONS computer program establishes an environment within which students solve problems. Each problem is presented in the same form, and that common form creates the problem solving environment. Each problem presented requires the student to infer a rule which

BEST COPY AVAILABLE

transforms one number into another number so that in each problem the student infers a new rule.

The student's task is to infer the rule from examples. To aid this process, the student can type in numbers for the computer to transform or he can test his understanding by transforming numbers the computer chooses and receiving feedback about his answers. When the student completes work on one rule, he can change to a new problem. The next subsection describes how the structure of the program encourages good problem solving.

The concept of a function is fundamental to mathematics. Throughout mathematics, rules which relate one symbol to another are basic content. Much mathematics work focuses on invention, recognition, generalization, and description of functional relations between sets of numbers. One-fourth of the objectives of elementary school mathematics requires the student to learn or extend a functional rule.

Since the problem solving program requires all problems to be functional relationships, one-fourth of elementary school content was available for teaching. Each objective to be taught required a sequence of problems carefully designed to help students induce appropriate rules. Preparation of an objective for presentation via the computer program involved specifying the sequence of problems through detailed task analysis. The second subsection describes the method of analysis and gives an extended example.

The MATH FUNCTIONS program was introduced into an elementary school for purposes of formative evaluation in Spring 1972. The program was fully integrated with standard operating procedures at the school, and was used freely by the students. In Fall 1972 more objectives were added, and formative evaluation continued on a full school basis. The third subsection describes how the program was introduced to the school and the formative evaluation procedures used to modify the program.

BEST COPY AVAILABLE

B. The FUNCTIONS Program

The FUNCTIONS program was designed to create an environment in which students develop problem solving skills. Features of the environment encourage students to state the problem clearly, gather data, organize data, use feedback, formulate and test hypotheses, decide when they have finished, divide complicated problems into subproblems, and integrate subsolutions into full solutions. This subsection examines the interactions with the student, the kinds of problems presented by FUNCTIONS, and the features of the program which teach specific problem solving skills.

1. The Interactions with the Student

Every time a student uses FUNCTIONS, he solves a problem of the same kind. Each problem requires him to infer a rule by which certain numbers or letters (the domain elements) are transformed into other numbers or letters (the range elements). The student must induce the rule from particular examples of correct domain-range pairs which are displayed for him. For example, if the rule is "add six to the domain element," the student would initially see a sample display like that in Figure 1. Notice that the domain elements (6 and 34) are displayed on the left of the word "BECOMES" and the corresponding range elements (12 and 40) are displayed on the right. This form is maintained for all problems displayed by the FUNCTIONS program.

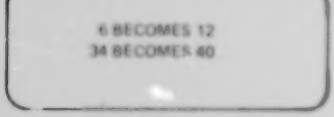


Figure 1. The sample display

BEST COPY AVAILABLE

The Input Mode. Usually the first samples of the problem are insufficient for the student to induce the rule. The student can generate more data by using the input mode of the program. During input mode, the student types any domain element he wishes, and the program transforms the domain element into a range element and displays the resulting pair for the student. The input mode display appears in Figure 2. Notice that the domain is specified for the student as "A NUMERAL BETWEEN 1 and 99." The three stars (***) tell the student that he is to type something. In Figure 2 the student typed "7." The display after a correct input appears in Figure 3.

```
 6 BECOMES 12
34 BECOMES 40

TYPE A NUMERAL BETWEEN 1 AND 99
OR "TEST" OR "CHANGE"
*** 7
```

Figure 2. The input mode display

```
 6 BECOMES 12
34 BECOMES 40
 7 BECOMES 13

TYPE A NUMERAL BETWEEN 1 AND 99
OR "TEST" OR "CHANGE"
*** 100
```

Figure 3. A correct input display

BEST COPY AVAILABLE

Input mode remains in effect after inputs, so the student may type many inputs in a row. In Figure 3 the student tries to input 100, a number too large for the domain, so the program produces an error message instead of a range element. The incorrect input display appears in Figure 4.

```
 6 BECOMES 12
34 BECOMES 40
 7 BECOMES 13
100 IS TOO LONG

TYPE A NUMERAL BETWEEN 1 AND 99
OR "TEST" OR "CHANGE"
*** TEST
```

Figure 4 An incorrect input display

The student can continue typing inputs as long as he wishes and he will continue to get corrective feedback or new domain-range pairs. When the student is ready, he can enter the test mode by typing the word "TEST," as shown in Figure 4.

The Test Mode. In the test mode of FUNCTIONS, the program supplies domain elements for the student to process according to his understanding of the rule. A display in test mode appears in Figure 5. A correct answer, like that given in Figure 5, is followed by the sound of a bell and the display in Figure 6.

BEST COPY AVAILABLE

```
 6 BECOMES 12
34 BECOMES 40
 7 BECOMES 13

TYPE "CHANGE" OR THE ANSWER TO
21 BECOMES
*** 27
```

Figure 5. The test mode display

```
 6 BECOMES 12
34 BECOMES 40
 7 BECOMES 13
21 BECOMES 27

GREAT!

TYPE "CHANGE" OR THE ANSWER TO
97 BECOMES
***913
```

Figure 6. A correct test display

Students remain in the test mode as long as tests are answered correctly. The new test (97) in Figure 6 was created since the last answer given was correct. The student erroneously responds to 97 with 913. This error leads to the incorrect test display in Figure 7; the form of the display with the error directly under the correct answer facilitates comparison. Notice that an error in the test mode transfers control to the input mode, a feature designed to encourage the student to explore

BEST COPY AVAILABLE

inputs related to the domain element which caused his error; in this case, numbers near 97 would be relevant inputs.

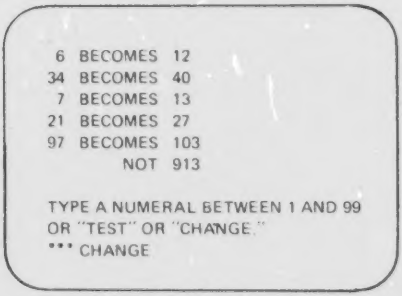


Figure 7. An incorrect test display

The Change Option. The option to change problems is available in both test and input modes. When the student types "CHANGE," the FUNCTIONS program prepares another problem, presents the sample display and initiates the input mode. The student can type "CHANGE" whenever he wishes. However, if he has not taken enough tests to demonstrate his knowledge (or lack of knowledge) of the rules, he will not be allowed to see a new problem. He does not need to master the rule, but he must try at least one test. Suppose the student has taken no tests, and types "CHANGE." The computer displays the message, "YOU MUST TAKE MORE TESTS BEFORE TYPING 'CHANGE,'" shown in Figure 8.

BEST COPY AVAILABLE

```
 6 BECOMES 12
34 BECOMES 40
 7 BECOMES 13
21 BECOMES 27
97 BECOMES 103

YOU MUST TAKE MORE TESTS BEFORE TYPING "CHANGE."
TYPE "CHANGE" OR THE ANSWER TO
56 BECOMES
•••
```

Figure 8. An incorrect change display

Special Messages. Two additional messages provide guidance to the student. After six or more inputs in a row, he is encouraged, but not required, to enter the test mode. This feature urges the student to develop a hypothesis to test within six inputs, and to proceed to test his hypothesis. Assuming 99 was the sixth successive input, the too-many-input display would appear as shown in Figure 9.

BEST COPY AVAILABLE

```
 6 BECOMES 12
34 BECOMES 40
 7 BECOMES 13
21 BECOMES 27
97 BECOMES 103
54 BECOMES 60
94 BECOMES 100
95 BECOMES 101
99 BECOMES 105

PLEASE TYPE "TEST" SOON.
TYPE A NUMERAL BETWEEN 1 AND 99
OR "TEST" OR "CHANGE"
***
```

Figure 9. The too-many-input display

Similarly, after six correct tests in a row, the student has, in most cases, demonstrated sufficient mastery of the rule and is encouraged to change to a new problem. Assuming now that 99 is the sixth correct test in a row, the too-many-tests display would appear as in Figure 10.

```
 6 BECOMES 12
34 BECOMES 40
 7 BECOMES 13
21 BECOMES 27
97 BECOMES 103
54 BECOMES 60
94 BECOMES 100
95 BECOMES 101
99 BECOMES 105
GREAT!

PLEASE TYPE "CHANGE" SOON.
TYPE "CHANGE" OR THE ANSWER TO
29 BECOMES
***
```

Figure 10. The too-many-tests display

BEST COPY AVAILABLE

Summary of Interaction with the Student. The last pages presented an example of a student interacting with the FUNCTIONS program to solve a problem. In this example, the student's problem was to find a rule which explains how numbers between 1 and 99, the domain elements, are transformed to produce the corresponding range elements. The input mode and the test mode are designed to help him. The student can type as many inputs as he wishes in input mode. Each input is examined for legality. Illegal inputs are rejected, with an explanation for the rejection. Legal domain elements are transformed and displayed with their corresponding range elements. The display is organized for easy reference. In test mode, which the students enters by typing "TEST," the computer provides domain elements which the student transforms. Student answers are evaluated and feedback is provided. On incorrect answers, both the correct answer and the student's answer are displayed. On correct answers, the student is reinforced by a bell sound and the word "GREAT!" Students may decide how many tests to take. There is no limit to the number of times the student can use both test and input modes; and the student can terminate his own work on the problem by typing "CHANGE." Special messages are displayed if the student tries to change problems without self-testing, if he makes too many inputs, and if he takes too many tests in a row.

2. The Kind of Problem Presented

An enormous variety of problems can be presented with the FUNCTIONS program. This subsection describes the three properties common to all problems used with FUNCTIONS and gives some representative examples. The three common properties are: (a) the rule must be a mathematical function; (b) the range and domain must be composed of alphabetic or numeric characters; and (c) the rule, the domain, and the range must be simple.

BEST COPY AVAILABLE

The Three Properties Common to All FUNCTIONS Problems. All problems that the FUNCTIONS program presents are mathematical functions, a fact recognized in the name of the program. Mathematically, a function is described by the domain set, the range set and the rule by which domain elements are transformed into range elements. Rules in functions are restricted, by definition, to those which transform each element of the domain into a single element of the range. Some examples will clarify the definition of function. The rule, "Add one to the domain element," is a legitimate functional rule since each domain number is transformed into one range number. The rule, "If the domain number is even, say 'EVEN'; if it is odd, say 'ODD,'" is also a legitimate functional rule, since each domain number is transformed to exactly one range element. By contrast, the rule, "Take the square root of the domain element," is not a functional rule, since each positive domain element has two corresponding range elements.

Within the context of the FUNCTIONS program, the mathematical definition of function assumes an important role. Each display line contains one pair of domain-range elements connected by the functional rule. The student knows that pairs are fixed for all time. If he repeats a domain element in input mode, he will see an identical pair. This restriction simplifies the search for a rule.

All pairs of domain and range elements the FUNCTIONS program presents are composed of alphabetic, numeric, and certain special characters contained on a standard typewriter. The length of the domain element plus the length of the range element must be less than 72 characters. A single line of display contains a maximum of 72 characters, and instructional considerations dictate having exactly one display line per pair. If more than one line per pair were allowed, the density of information in the display would be reduced and the student would be required

BEST COPY AVAILABLE

to type too much to get a single display. Both of these could lead to a deterioration of performance.

The domain, range, and rule must be simple for all problems the FUNCTIONS program presents. Simple sets have easily recognized elements and can be described in a few words; complex sets fail one or both of these criteria. The concept is best illustrated by examples. Some simple sets are: (a) numbers from 1 to 99, (b) even numbers, (c) three to five letters, (d) two numbers separated by a space, and (e) a fraction. Some complex sets are: (a) numbers which have three or more factors, and (b) a digit followed by at least as many letters as that digit signifies.

Both the domain and the range must be simple sets for instructional reasons. The student should not spend much of his time trying to understand the domain set or trying to create legal inputs that would pull his attention away from the crucial task of inferring the rule. The domain set is described explicitly in the input mode display for the problem, so it must have a simple verbal description. In the example in Figure 2, the description of the domain was "A NUMERAL BETWEEN 1 AND 99."

The rules that the FUNCTIONS program presents must also be simple. A simple rule can be described easily in words and contains no arbitrary exceptions. As with simple sets, examples will best illustrate the meaning of simple rules (see Table 1).

BEST COPY AVAILABLE

TABLE 1

Examples of Simple and Complex Rules

| PROBLEM | |
|---|---|
| SIMPLE | EXAMPLE |
| Even numbers become "EVEN", odd numbers become "ODD." | 2026 BECOMES EVEN<br>21 BECOMES ODD |
| Each number becomes itself plus 6. | 12 BECOMES 18<br>314 BECOMES 320 |
| Each domain element becomes itself in reverse order. | 2341 BECOMES 1432<br>RAT BECOMES TAR |
| Each domain element becomes the number of characters in the element. | AXBR BECOMES 1<br>12 BECOMES 2 |
| Each domain element becomes its left most character. | 6134 BECOMES 6<br>941 BECOMES 9 |
| COMPLEX | |
| Every number becomes twice itself except that 21 BECOMES 12. | 504 BECOMES 1008<br>21 BECOMES 12 |
| Every number becomes the number of characters in its corresponding Roman Numeral | 312 BECOMES 6<br>88 BECOMES 8 |

The simplicity of a rule is a relative concept depending on the knowledge of other rules the student brings to the problem. For most adults, long division is a simple rule; for a student who has not yet learned it, long division is complex. This point will be illustrated at length in the section on how the math curriculum was prepared for presentation. Special procedures were developed to simplify complex rules by teaching appropriate preliminary rules.

The rationale for requiring simple rules is motivational. Experience working in FUNCTIONS must lead to positive feelings or people will not seek the experience. People will be frustrated by complex rules, with arbitrary exceptions, so that simple rules will be easier and more satisfying to solve.

BEST COPY AVAILABLE

Some Examples of Problems. More than eight hundred problems are available for presentation by the FUNCTIONS program. A selection from the problems is contained in Table 2. Each problem is described by its domain, its range, and the verbal statement of the rule. Examples of each problem accompany the description.

TABLE 2

Examples of FUNCTIONS Puzzles

| PROBLEM | EXAMPLES |
|---|---|
| Domain: Numbers 1 to 99<br>Range: Numbers 7 to 105<br>Rule: Domain element is increased by 6 | 6 BECOMES 12<br>91 BECOMES 97<br>21 BECOMES 27 |
| Domain: 6 to 11 letters<br>Range: Single letters<br>Rule: Second letter of the domain is the range element | GMRSYV BECOMES M<br>TLMDYVTR BECOMES L<br>ABCEDFGH BECOMES B |
| Domain: Two numerals separated by space<br>Range: The signs >, <, =<br>Rule: If the two numbers are equal, =; if the first is greater than the second, >; if the first is less than the second, < | 28 43 BECOMES <<br>264 26 BECOMES ><br>462 461 BECOMES ><br>34 34 BECOMES = |

BEST COPY AVAILABLE

TABLE 2 (cont'd)

Examples of FUNCTIONS Puzzles

| PROBLEM | | EXAMPLES |
|---|---|---|
| Domain: | A number 1 to 999 | 4 BECOMES 24<br>14 BECOMES 54<br>114 BECOMES 354 |
| Range: | A number 1 to 9999 | |
| Rule: | Domain element is multiplied by 3, then 12 is added | |
| Domain: | A digit, then some letters | 2ABMC BECOMES B<br>9RMX BECOMES NO<br>7RAXM BECOMES NO<br>4TLCDR BECOMES D |
| Range: | All letters and word "NO" | |
| Rule: | Let N equal the digit. If there are at least N letters, answer is the Nth letter. Otherwise "NO." | |
| Domain: | 1 to 5 letters | ABC BECOMES 3<br>TRMX BECOMES 4<br>CLMWQ BECOMES 5 |
| Range: | Digits | |
| Rule: | The number of letters in the domain element | |
| Domain: | 1 to 18 letters | RMXLTV BECOMES 6<br>QRSTUMXLT BECOMES 9<br>ABM BECOMES 2 |
| Range: | Numbers 1 to 18 | |
| Rule: | The number of letters in the domain element | |
| Domain: | 2 letters | PU BECOMES PV<br>EL BECOMES EM<br>ON BECOMES OO<br>UN BECOMES UO<br>WL BECOMES WM |
| Range: | 2 letters | |
| Rule: | The first letter of the range is the first letter of the domain. The second letter of the range is the letter of the alphabet after the second letter of the domain | |
| Domain: | 1 to 9 letters | ARMX BECOMES XMRA<br>TUWLTRV BECOMES VRTLWUT<br>MABCBAM BECOMES MABCBAM |
| Range: | 1 to 9 letters | |
| Rule: | The domain element is written backwards | |
| Domain: | 1 to 9 digits | 9185 BECOMES 98<br>21387 BECOMES 237<br>2222222 BECOMES 2222 |
| Range: | 1 to 5 digits | |
| Rule: | Range is every other digit of domain element, starting with the first | |

BEST COPY AVAILABLE

### 5. Problem Solving and FUNCTIONS

The last subsections described the interaction of the student with the FUNCTIONS program and the kinds of problems which are presented by the program. This subsection describes how each component of the FUNCTIONS program was designed to teach problem solving skills by the structure it imposes on the interaction. Design decisions are always based on assumptions. We hypothesize that the structural features of FUNCTIONS described in the sections that follow will, in fact, teach the skills they are designed to teach. A forthcoming paper will relate empirical findings to these hypotheses. Eight skills comprise the model of problem solving for FUNCTIONS; this subsection shows precisely which interaction features were designed to teach each component skill. The eight skills are: (a) stating the problem clearly, (b) gathering data, (c) organizing data, (d) using feedback, (e) formulating and testing hypotheses, (f) knowing you have finished, (g) dividing a problem into subproblems, and (h) integrating subsolutions into a full solution.

Stating the Problem Clearly. Stating the problem clearly is essential to solving any problem. Whenever a student works with the FUNCTIONS program, he knows that he must induce a rule from the examples provided. He knows the two modes of interaction (input mode and test mode) which he can use to help him solve the problem. This regularity of the task is imposed by the structure of the program. The specific rule used in a problem is not important in defining the problem. In fact, if the student knew the rule, he would no longer have a problem to solve.

Gathering Data. Gathering data is crucial to problem solving. Both interaction modes in the FUNCTIONS program cause data to accumulate. The input mode gives specifically requested data. The test mode

BEST COPY AVAILABLE

gives randomly generated data from within the domain set. The program emphasizes the importance of data gathering for problem solving by continually increasing the store of data.

The two modes for gathering data require different activities from the student. Input mode requires active data seeking with a passive response; test mode requires passive data receiving with an active response. In input mode, the student must create his own domain element, to help him explore a specific hypothesis about the rule. Formulation of the domain element is an active process akin to designing an experiment in science. When the computer processes the domain element and displays the corresponding range element, the student can take in the data with no further overt response. In test mode, the student is given the domain element by the computer. He must then process it according to the rule, as he has formulated it, and produce a range element. The computer judges the response, and displays the correct answer with appropriate feedback.

Both methods of data gathering are common. The FUNCTIONS program makes the two modes explicit and allows the student to try both. Students may be more comfortable using input mode; others may prefer test mode for data gathering. Neither method is ideal; the students using FUNCTIONS have the opportunity to determine and exercise the style of data gathering best suited to their individual needs.

Organizing Data. Organizing data makes extracting information easier. The tabular form in which the domain-range pairs are displayed and the parallel form in which test errors are displayed are two features of FUNCTIONS which organize the data for the student.

The overall tabular form of the display places one complete domain range pair on each line. None of the irrelevant, repetitive text such as

BEST COPY AVAILABLE

positive feedback on correct tests or the input mode prompts are contained in the main display. Such irrelevant text would reduce the density of information and make the display less useful. The columnar arrangement of the domain and range elements facilitates scanning the display and makes it easier to extract information. The word "BECOMES," displayed on each line, separates the domain from the range elements and further organizes the display. (See Figure 11.)

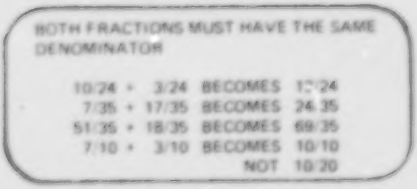
BOTH FRACTIONS MUST HAVE THE SAME DENOMINATOR
10/24 + 3/24 BECOMES 13/24
7/35 + 17/35 BECOMES 24/35
51/35 + 18/35 BECOMES 69/35
7/10 + 3/10 BECOMES 10/10
NOT 10/20

Figure 11 The tabular form of the display

The parallel form in which test errors are displayed facilitates character by character comparison of the correct answer with the student's answer. In Figure 11, the student could easily notice that the numerators of the two answers were the same, and isolate his difficulty as the difference between the two denominators (10 and 20). This would then focus his attention on figuring out how to get the correct denominator when adding two fractions.

In addition to the two program imposed data organizing devices, the program allows the student to further structure his data by the inputs he creates. When the inputs are numeric, putting in sequential numbers can create an organized display which greatly aids problem solving. When

BEST COPY AVAILABLE

the input is alphabetic, systematic inputs such as the alphabet, or a string of identical letters often produce especially useful displays. We expect students to discover this powerful way of using the input mode.

Using Feedback. Using feedback is essential to problem solving. When feedback is clear, as it is in the FUNCTIONS program, the student can learn to use it. Feedback in FUNCTIONS occurs in several ways, all of which were described before.

In input mode, the program responds to suggested domain elements with appropriate feedback. If the suggested domain element is legitimate, the program produces a correct input display showing the student one new domain-range pair. (See Figure 3.) If the suggested domain element is illegitimate, the program produces an incorrect input display, telling the student why his element is incorrect. (See Figure 4.) When the student has stayed in the input mode too long, he is requested to change to the test mode. (See Figure 5.) The request is an important feature; the program could require the student to change. The distinction is crucial: by allowing the student a choice, the program gives him an opportunity to respond to feedback. By requiring the student to change, the program would force him into passive acceptance.

In test mode, the program feeds back the correct answer and a congratulatory message (GREAT!) when the student is correct. (See Figure 6.) When the student is incorrect, the program displays the correct answer and his answer in a format designed to facilitate comparing the answers. (See Figure 7.)

On incorrect answers the student is automatically switched to input mode. Once he begins testing himself, he can only return to input mode by making an error. The switch in modes is feedback to the student, telling him that after errors, inputs are likely to be helpful. Again, as

BEST COPY AVAILABLE

with other feedback from the program, the student may ignore the feedback and transfer back to the test mode. After six correct tests in a row, the student is requested to change to a new problem. (See Figure 10.) Again, the request allows the student an opportunity to respond to feedback rather than forcing him into passive acceptance.

The final feedback message concerns incorrect use of the change option. The student may not change puzzles without testing himself. (See Figure 8.) Feedback on incorrect use of change is coercive; the student cannot ignore it. The reason for this one exception to the general ban on coercive feedback is technical. The FUNCTIONS program gets all of its information about student understanding from test mode responses. If there are no tests, the program cannot decide what the student should do next, therefore, at least one test is required.

Formulating and Testing Hypotheses. Formulating and testing hypotheses is an important problem solving skill. The FUNCTIONS program was designed to force the student to engage in hypothesis formation and testing. In the input mode, some hypothesis guides the choice of domain elements. By his inputs, the student can refine his emerging theory. Most students predict the outcome of an input before the computer displays it and check their implicit prediction against the explicit outcome. In test mode, the student must actively use his hypotheses to create an answer. The computer feedback on the answer tells the student if his answer was correct, and he can use that information to further refine his hypothesis or to strengthen his belief in it. When the student gives an incorrect answer, he is returned to the input mode. (See Figure 7.) This change reflects the necessity of refining the current hypotheses, and suggests that further inputs may help.

BEST COPY AVAILABLE

Knowing You have Finished. Knowing that you have finished a problem is sometimes difficult. In FUNCTIONS, the student can never be certain that he understands the rule completely, no matter how many correct predictions he makes on tests. Nonetheless, as the number of correct test items increases, the probability that the hypothesized rule is incomplete rapidly approaches zero. All puzzles were designed so that five correct tests in a row can occur by chance less than 5 percent of the time. For most puzzles, three correct tests in a row can occur by chance less than 1 percent of the time. Knowing when you have finished a FUNCTIONS puzzle is equivalent to recognizing when the chances you are just guessing correctly become sufficiently small, or when the chances you will ever succeed on the puzzle become too small. Students should type "CHANGE" when they have correctly answered several tests in a row, or when they have failed to answer a test correctly, and have no idea about how to proceed. We expect that students will learn to recognize these two conditions, and use CHANGE to begin a new puzzle. Two features are designed to help: the request to "PLEASE TYPE 'CHANGE' SOON" when the student has six or more correct tests in a row, and the message "YOU MUST TAKE MORE TESTS BEFORE TYPING 'CHANGE' " when the student has not taken enough tests to demonstrate mastery.

Subdividing Problems and Integrating Subsolutions. Dividing a problem into subproblems and integrating subsolutions into full solutions are effective heuristic strategies for solving problems. The FUNCTIONS program models the strategies and provides opportunity for the student to practice them. Modeling occurs across problems rather than within one problem. The sequence of problems presented requires the student to build more and more complex rules from those he already knows. The subsection on preparing the mathematical content for use with the FUNCTIONS program illustrates this point in detail, and the reader is referred to that discussion.

BEST COPY AVAILABLE

Opportunity to practice dividing a problem into subproblems occurs in each problem. It is often easier to formulate a rule for some particular class of elements within the domain set than to formulate an all inclusive rule. An example will clarify the point.

Consider the display in Figure 12. The first three domain range pairs were presented as samples. The student typed in two inputs (1111 and 2222) to determine if single digits were being transformed. In this case, the inputs were extremely helpful; each digit 1 is transformed to 8, each 2 to 3. Dividing the original problem into ten subproblems makes the problem tractable. The full rule requires each digit to be transformed into its arbitrarily selected partner. Integrating the ten partial rules into the full rule creates the final solution.

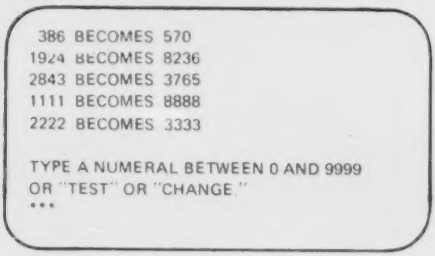


Figure 12. Display illustrating subdividing problems

Summary of Problem Solving and FUNCTIONS. This subsection examined how the FUNCTIONS program teaches problem solving skills through its structure. The structure limits what the student can do and focuses his energy into productive problem solving activities. Each feature of the program is included for its contribution to teaching of problem

BEST COPY AVAILABLE

solving. Eight skills were examined and discussed in light of the particular program features designed to teach them.

C. Preparation of Mathematical Content

This section first describes where functions occur in elementary school mathematics curricula in general and how the Individualized Mathematics system includes features making it especially adaptable for teaching with the FUNCTIONS program. The next subsection details the method of analyzing objectives and creating teaching sequences for presentation with the FUNCTIONS program, taking into account the fact that every puzzle presented must be a function with simple domain and range sets, and a simple rule. The section closes with an extended example of how one specific objective is taught using FUNCTIONS. The example illustrates how every objective is taught.

1. Functions in Elementary School Mathematics

The abstract, mathematical concept of function becomes concrete in elementary school mathematics in the form of computational skills and a variety of other skills such as counting and naming numbers. All elementary school mathematics curricula emphasize skills based on learning and extending functions.

To see clearly how a specific skill corresponds to a function, you must identify the domain set (the things which are transformed), the range set (the things which domain elements become when transformed), and the rule which is used in the transformation. Several examples will illustrate how to do this.

In most elementary school mathematics curricula, students learn to look at a numeral written in standard notation and decide whether the number is even or odd. What is the domain set? The things transformed

BEST COPY AVAILABLE

are standard numerals for whole numbers like 64, 13, 86, 934, and 2167. What is the range set? The things the domain elements become are the words "EVEN" and "ODD." What is the rule? Several different rules are adequate; all of them are equivalent. One such rule is: "Examine the right-most digit of the domain element. If it is an element of the set $\{0, 2, 4, 6, 8\}$, then the corresponding range element is 'EVEN'; otherwise, the corresponding range element is 'ODD.'" Notice that each range element corresponds to many domain elements, but each domain element corresponds to only one of the two range elements.

As a second example, consider the skill of adding two single digit numbers. All elementary school mathematics curricula require students to master this skill, often at several different levels. Different behavior is adequate for mastery at different times. Initially, counting with blocks or fingers might be enough; later, immediate recall without aids is required. To be concrete, assume the student must use an addition table to produce his answer. What is the domain set? The things transformed are pairs of numbers, each of the two numbers is between 0 and 9. Domain elements are represented as "6 + 4" or "9 + 0" or "3 + 4." What is the range set? The things the domain elements become are the numbers between 0 and 18. What is the rule? One possible rule is: "Look for the first number on the top of a column in the addition table; look for the second number at the left of a row; find the number at the intersection of the column and row and repeat that number." Each pair in the domain corresponds to one range element. This rule might be an extension of previously learned rules about tables or a completely new rule, depending upon the specific curriculum in which it is embedded.

The third and final example is the skill of determining which of two numbers is larger. Assume the two numbers are represented as

BEST COPY AVAILABLE

standard numerals. The domain set is pairs of numbers again, each number between 0 and 999. The range set is the three signs for greater than, equal to, and less than: {>, =, <}. The rule is: "If the two numerals are identical, the numbers are equal; if one is longer than the other, it is greater. If both are the same length, match digit-by-digit from the left until one has a larger digit; that number is greater."

All elementary school mathematics curricula include the three functions discussed above. In addition, most contain other functions involving the basic calculation algorithms of addition, subtraction, multiplication, and division of whole numbers. Many curricula also teach counting of objects or symbols, Roman numerals, word names for numbers, place values, and rounding off. These and other concepts can be viewed as instances of the abstract concept of functions and are, therefore, possible concepts to teach using the FUNCTIONS program.

2. The Individualized Mathematics System

While every elementary school mathematics curriculum contains substantial portions that could be taught with FUNCTIONS, the Individualized Mathematics system (Lindvall & Bolvin, 1966) developed at the Learning Research and Development Center has characteristics which make it particularly suitable for our purposes. The desirable characteristics of Individualized Mathematics include the individualized, self-paced instruction, the behavioral specification of content, the guaranteed mastery learning of all skills, and the clear statement of the prerequisite skills for each new skill. The next paragraphs clarify the reasons these characteristics are desirable.

The Individualized Mathematics system requires each student to learn each skill in the curriculum. Instruction is always aimed at individual students rather than at groups of students. No one ever advances

BEST COPY AVAILABLE

because others advance; each student learns independently. The individualized character of all instruction in the system makes it particularly suited to individualized instruction on the computer. The self-pacing character of the curriculum means the student can exercise more control over his instruction than he could in more rigidly paced systems.

All content in the Individualized Mathematics system is specified behaviorally. Behavioral specification makes clear exactly what the student must do to demonstrate mastery. Each objective in the curriculum is a specific step forward in the student's knowledge, and it can be taught in a variety of ways. Behavioral specification makes it possible to delineate precisely what domain set, range set, and rule must be taught for each objective.

Mastery learning is fundamental to the Individualized Mathematics system. Each student takes a test on each objective. He must pass with more than 85 percent correct before he is allowed to progress to the next skill. In this manner, the system assures that each student knows each skill before he advances to the next skill. No one advances simply because the rest of his class is ready to advance. Each student masters each skill in the curriculum.

Finally, the Individualized Mathematics curriculum clarifies what the student knows before he begins his study of a new objective. He has demonstrated mastery of each skill before the current skill, and instruction is built on the assumption that he remembers most easier skills as defined by the curriculum sequence. This assumption offers an enormous advantage over less structured curricula in which prerequisite behaviors are unclear. Instruction can be far less branched and complex when entry skills are known.

BEST COPY AVAILABLE

The Individualized Mathematics system contains approximately four hundred objectives for kindergarten through sixth grade. One hundred sixty of those are concerned with functions. Most other mathematics curricula would have a similar percentage of their content concerned explicitly with functions. In the discussion that follows, examples are taken exclusively from the Individualized Mathematics system.

3. Creating Teaching Sequences

The previous subsection demonstrated that a significant fraction of the skills in the Individualized Mathematics curriculum requires the student to learn or generalize functions. This subsection describes how teaching sequences for such skills are created for use with the MATH FUNCTIONS program.

Overview of Creating Teaching Sequences. The MATH FUNCTIONS program can teach skills by presenting an ordered sequence of puzzles beginning with a rule the student already knows and progressing by small steps to the criterion performance. Each puzzle in the sequence must be only slightly more difficult than the previous puzzle so that the student can induce the rule. If great gaps in difficulty exist, students will fail to induce the rules and fail to learn the criterion performance. A sequence of five puzzles is sufficient to teach most objectives. This section describes how objectives are analyzed and instructional sequences are created. The next subsection provides a detailed example for one objective.

Five steps are required to describe a teaching sequence once an objective has been selected. First, the criterion behavior and the behaviors the student is assumed to possess must be described in precise terms. Second, a puzzle which requires the criterion skill must be designed. Careful attention is given to the domain, range, and rule to

BEST COPY AVAILABLE

make them as close as possible to the corresponding domain, range, and rule on the paper and pencil form. Third, a sequence of instructional puzzles which proceeds by small increments of complexity from the entry behavior to the criterion behavior is designed. This instructional sequence does the teaching in most cases. However, if the student fails to solve the puzzle at entry level, he must have remedial material. The fourth step in preparing a teaching sequence is to create a remedial sequence of puzzles which every student should be able to solve. Fifth, a sequence of extensions beyond the criterion behavior is prepared for students who wish to generalize the required behavior. The rest of this subsection examines each step in turn.

The Criterion and Prerequisite Performances. The first step in preparing to teach a mathematical objective using the FUNCTIONS program is to state as clearly as possible what the student must do when he has learned the objective and what he already can do. Details, which might be irrelevant to a human teacher, are essential to the computer program. Consider an example: Suppose the student must learn to add. Much is left unclear. How many members must he add at once? What kind of numbers are involved--whole, negative, fractional, decimal, or imaginary? How large are the numbers? The clearer statement, "Given two 2-digit positive integers, the student will add them," is sufficient for most purposes. However, for a computer program, more must be specified: Should the student have to regroup (carry) from the ones place to the tens place? Should the sums be limited to 2-digit numbers as the addends are? One complete statement of the objective would be: "Given two 2-digit positive integers, the student will add them. The sum must be a 2-digit integer. Regrouping should occur in the ones' place in half the test items." All the details of the criterion performance are specified before any instructional puzzles are designed.

BEST COPY AVAILABLE

For the same reasons the criterion performance is precisely specified, the prerequisite performances must also be specified precisely. Prerequisite performances will be simple for the student and, therefore, can serve as introductory puzzles for the lesson.

The Criterion Puzzle. The second step in preparing a teaching sequence is to create a domain, range, and rule which match the criterion performance as closely as possible. Certain features, important to the problem solving aspects of FUNCTIONS, restrict the choice of domain, range, and rule. The domain and range elements must be short, fitting on one line of an alphabetic or numeric display and the sets must be simple, since complex sets require too much attention of the student. Rules must also be simple relative to the student's prerequisite skills. For most objectives, creating the criterion puzzle is a straightforward exercise once the objective has been precisely stated. For the objective stated in step one, the domain set would be "two numerals between 10 and 99 separated by '+' whose sum is less than 100"; the range would be "a numeral between 20 and 99"; and the rule would be "add using the standard algorithm." Typical samples for this puzzle appear in Figure 13.

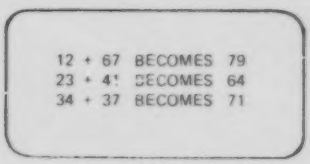


Figure 13 Sample display for criterion puzzle

For a few objectives, the domain of the criterion puzzle is unavoidably complex, making it impossible to meet all the restrictions on puzzles simultaneously. In such cases, the student is taught to understand the

BEST COPY AVAILABLE

complex domain in small steps just as he is taught the complex rule. The extended example in the next section describes a teaching sequence for an objective with a complex domain.

The Main Instructional Sequence. The third step in preparing a teaching sequence is to create a sequence of puzzles which begins with an entry performance and ends with the criterion performance. Each puzzle differs from its immediate predecessor by a small increment in complexity, either in the rule or in the domain. Creating such sequences is something of an art, although there are some clear guidelines for what makes puzzles more complex. The rest of this section contains a warning about possible pitfalls, and then presents some rules and examples to illustrate how instructional sequences can be created.

The Overall Strategy and Its Difficulties. To make a puzzle more difficult, you can normally change the domain, the range, or the rule, but not more than one at a time. However, there is no infallible way to create two puzzles which differ by a small increment in complexity. Some seemingly trivial changes cause great differences in puzzle complexity. To a large extent, the complexity of a puzzle depends on what the student knows, and the author of a sequence must place himself in the student's mind as best as he can.

In order to clarify the point, consider the following example. Suppose two rules operate on the same domain and range: Numbers from 1 to 100 are transformed to single digit numbers. The rules are slightly different in case one, the single digit is the remainder on division by nine; in the second case, the single digit is the remainder on division by ten. The rules differ only in one word and naively seem about the same difficulty; however, "the remainder on division by ten" is equivalent to "the last digit of the number," and students will invariably solve the

BEST COPY AVAILABLE

second rule in this simpler fashion. In less obvious cases, it takes considerable imagination to anticipate how students, or even other adults, will solve puzzles. Frequently, different people solve puzzles with completely different rules.

The Four Ways to Make Puzzles More Complex. With the warning of the last section still in mind, let us consider four ways in which puzzles can be made slightly more complex: (a) the domain can be extended to larger numbers with the same rule in effect; (b) one more symbol can be added to the domain; (c) a rule can be generalized to a new domain; and (d) a rule can combine two previously mastered rules. The next paragraphs give examples of each type of change from actual teaching sequences.

Extending the domain to larger numbers with the same rule produces a more complex puzzle. In one objective, the student learns to solve open addition number sentences for the unknown "N." The addends and the sum are all whole numbers less than 99. In one puzzle, the addends are restricted to numbers less then 10, and the student learns to subtract to solve such number sentences as:

N + 6 = 11 BECOMES 5
5 + N = 7 BECOMES 2

In the next more difficult problem, the addends are allowed to be as large as 99, with sums also up to 99. The same rule is executed on larger numbers.

As a second example, students learn to round off numbers to the tens' place for numbers up to 100 in problems such as:

21 BECOMES 20
37 BECOMES 40
9 BECOMES 10

BEST COPY AVAILABLE

The next puzzle is more complex, because the numbers are allowed to be as large as 999. The next puzzle would again extend the domain to numbers as large as 9999, yielding problems such as:

2158 BECOMES 2160
3472 BECOMES 3470
261 BECOMES 260
24 BECOMES 20

Adding one symbol to the domain makes puzzles more complex, since the relevant stimuli are masked by the new symbol. Consider again solving addition number sentences. An early puzzle requires the student to add the two numbers:

5 + 6 BECOMES 11
2 + 1 BECOMES 3
3 + 0 BECOMES 3

A more complex puzzle introduces the symbol "N" to the domain preparatory to moving the "N" to various positions:

N + 2 + 1 BECOMES 3
3 + 7 + N BECOMES 10

In rounding off objectives, the instructions about the place to round off to can be carried in a symbol before the actual number. The notation "10 124" tells the student to round 124 to the tens' place; while "100 124" tells him to round to the hundreds' place. Initially, the two number symbol is too complex, so it is introduced as an extension to a domain. First the student learns to round numbers up to 1000 to the tens' place.

231 BECOMES 230
624 BECOMES 620
937 BECOMES 940

In the next puzzle, the constant "10" is added to the domain element, while the same rule remains in effect:

BEST COPY AVAILABLE

10 231 BECOMES 230
10 624 BECOMES 620
10 937 BECOMES 940

Generalizing a known rule to a more extensive domain makes the puzzle more difficult. Generalization differs from simple extension to larger numbers in that the rule itself must be changed. Suppose a student has learned the Roman numerals up to five (I, II, III, IV, and V). His rule is probably quite particular: in fact, for 4 and 5, he has probably mastered a simple association. If he now must learn the Roman numerals to 10, he will need to generalize his rule. Further extensions to 50 and 100 will lead to a more generalized statement of the rules for changing Arabic numerals into Roman numerals. A second example of generalizing a known rule occurs when learning to add. Suppose the student knows how to add two 2-digit numbers to get sums up to 198. His rule may involve specific statements about the ones' and tens' places and not be general enough to allow him to add two 3-digit numbers reliably. Extending the puzzle to require him to add 3-digit numbers will force him to generalize his rule. A student who knows how to add in two base systems may be asked to generalize his knowledge to a third base system.

A puzzle in which the student must combine two previously mastered rules is more complex than a puzzle based on either rule alone would be. Suppose the student has learned to multiply fractions without reducing the result and has also learned to reduce a fraction to lowest terms. A puzzle which required the student to multiply and then reduce to produce the range element would be more complex than either rule alone. Suppose the student has learned to carry out additions of 2-digit numbers and has also learned to compare two numbers less than 100 for size:

BEST COPY AVAILABLE

21 ? 64 BECOMES <
21 ? 14 BECOMES >
14 ? 41 BECOMES <
12 ? 12 BECOMES =

The combined rule requiring him to add and then compare will be more complex:

20 + 16 ? 31 BECOMES >
12 ? 7 + 5 BECOMES =
61 + 12 ? 68 BECOMES >

In this case, the domain has also become more complex. These are often difficult to separate. A further increment in difficulty would be required if both sides of the "?" were problems:

21 + 16 ? 34 + 12 BECOMES <
17 + 23 ? 23 + 17 BECOMES =

More difficulty still can be introduced by allowing the operations in each problem to vary:

21 + 34 ? 34 - 21 BECOMES >
9 X 3 ? 12 + 13 BECOMES >
6 X 4 ? 38 - 16 BECOMES >

Combining previously mastered rules is the most common way to increase difficulty in a teaching sequence used with the FUNCTIONS program. A sequence of puzzles can be prepared by using one or more of the techniques for making puzzles slightly more complex. The sequence begins with skills the student is assumed to know, the entry performance, and proceeds in a few steps to the criterion performance. If each step is sufficiently small, the students will proceed easily to master the criterion performance. Some puzzles are designed to give remedial assistance because students sometimes forget what they already know.

The Remedial Sequence. The fourth step in creating a teaching sequence for an instructional objective is to produce a remedial sequence

BEST COPY AVAILABLE

of puzzles that every student, who has mastered the prerequisites, should be able to solve easily. The remedial sequence is used only by students who do not exhibit the entry performance in the instructional sequence. There are three reasons students fail entry performance: They are unfamiliar with the FUNCTIONS program; they forget one or more prerequisite skills; they do not recognize the domain elements as ones they know how to manipulate. The three causes of failure require differential consideration.

If the student is unfamiliar with the FUNCTIONS program, any additional work where the rule is easy should be helpful to him. He can "play around" on easier levels and learn to use the input mode, the test mode, and to change levels. No special puzzles are required.

If the student has forgotten one or more prerequisites, he may remember them with opportunity to practice. For such a student, puzzles requiring the prerequisite rules with no additional complexity are important. Remedial sequences include such puzzles.

For the student who has difficulty with the domain, a simpler puzzle whose domain contains one less symbol is helpful. A puzzle whose domain is identical, but whose rule is simpler, is equally likely to help. The student might be required to copy part of the domain to focus his attention on that part as a single symbol. For example, if the entry level for solving addition number sentences has a domain with many symbols such as "3 + 17 = N," and the student fails that level, a remedial level should maintain the domain, but change the rule to "copy the addition problem":

3 + 17 = N BECOMES 3 + 17
21 + 14 = N BECOMES 21 + 14
N = 16 + 34 BECOMES 16 + 34

BEST COPY AVAILABLE

Generally, the same methods apply to creating remedial sequences as apply to the main instructional sequences. The objective of the remedial sequence is to provide the student with the help he needs to begin work in the instructional sequence regardless of the cause of his difficulties.

The Post-Criterion Sequence. The final step in preparing a teaching sequence for a mathematical objective is to provide post-criterion extensions of the performance the student has learned. These additional puzzles are created according to the same principles as the main instructional sequence: Small increments in complexity are used to help the student generalize his rule and to apply the rule in new contexts. Post-criterion puzzles are optional for the students, but most choose to try at least one.

Summary of Creating Teaching Sequences. This subsection has set out general principles by which teaching sequences are created. Five steps are followed: (a) analyses and statement of the criterion and prerequisite performances; (b) description of the criterion performance for FUNCTIONS; (c) creation of a sequence of puzzles which increases by small steps from the entry performance to the criterion performance; (d) creation of a remedial sequence; and (5) creation of a post-criterion sequence. Teaching sequences were prepared for 120 objectives using these five steps. The next subsection describes how one of those 120 teaching sequences was created.

4. An Example of a Teaching Sequence

The last subsection described the general method for creating puzzles to teach mathematics content in the context of the FUNCTIONS program. This subsection contains a detailed example of the preparation of one objective. Each of the five steps outlined in the last subsection is illustrated.

BEST COPY AVAILABLE

Describing the Criterion and Prerequisite Performance. The objective, selected from the Individualized Mathematics system, requires the student to add two fractions, each less than one, with the same denominator and to state if the sum is less than, equal to, or greater than one. The student uses the standard signs ( $<$, $=$, and $>$) for less than, equal to, and greater than. The statement of the objective leaves some parameters unspecified. How large can the sum of fractions be? How large can the denominator be? Should the sum be reduced? The full specification of any objective in the Individualized Mathematics system includes the criterion-based, curriculum embedded test (CET), some items of which appear in Figure 14. All denominators on the CET are less than 100; the sum of the numerators is less than 100 and less than twice the denominator. Since the addends are not reduced to lowest terms (22/58 = 11/29), the sums need not be reduced.

Add the fractions. Then write $>$, $<$, or $=$ in the ☐ to show whether the sum is greater than, less than, or equal to 1.

$\frac{22}{58} + \frac{23}{58} =$ _____ The sum ☐ 1

$\frac{6}{11} + \frac{5}{11} =$ _____ The sum ☐ 1

$$\begin{array}{r} \frac{16}{34} \\ +\ \frac{25}{34} \\ \hline \end{array}$$

The sum ☐ 1

Figure 14. Criterion items for adding fractions and comparing the sum to one.

BEST COPY AVAILABLE

The criterion performance is now clear. What entry level skills are assured by the student's progress through Individualized Mathematics before he encounters this objective? The four nearest prerequisite skills in the curriculum include: (a) given two numbers less than 99, the student adds to obtain a sum less than 99; (b) given a specified denominator up to 10, the student writes a fraction equivalent to two wholes; (c) given an improper fraction between 1 and 2, the student rewrites it as a mixed fraction; and (d) given two numbers up to 99, the student compares them using the signs "<," "=," or ">" for less than, equal to, or greater than.

These four performances make the teaching task clearer. The student knows how to add the appropriate numbers; he must be taught which numbers to add. The student knows some fractions are larger than one; he must be taught to use that knowledge to assign the correct comparison. The student knows the concepts of greater than, less than, and equal to; he must be taught to apply them to fractions. The student knows how to compare numbers of the appropriate magnitude using the three signs; he must learn which numbers to compare.

Preparing a Criterion Puzzle. The second step in preparing a teaching sequence is to create a puzzle in FUNCTIONS format which approximates the criterion performance. Remember, there are three conditions which must be met: The domain and range elements must be displayed on a single line and must be made up of standard typewriter characters and certain special symbols; the length of the domain plus range elements must be less than 72 characters; the domain, range, and rule must all be simple relative to the student. The conditions do impose constraints on the puzzle in this case.

BEST COPY AVAILABLE

The curriculum embedded test (CET) items in Figure 14 use a multiple line symbol for a fraction. On the computer, a fraction must be represented on one line: "22/58." In the CET, both horizontal and vertical addition occur. On the computer, only horizontal addition fits on one line: "22/58 + 23/58." The student makes two distinct responses on the CET items: First he adds, then he compares. On the computer, the response format is similar: "45/58 <." A complete line for an item then appears as:

6/11 + 5/11 BECOMES 11/11 =
18/23 + 17/23 BECOMES 35/23 > .

This form satisfies all the constraints on line format: It is short, it is typeable, and it fits on a single line.

Figure 15 describes the range, domain, and rule for the criterion puzzle number 5, and for all the other puzzles in the teaching sequence. The range for the criterion puzzle matches the numeric constraints of the objective exactly. The domain also matches the numeric constraints of the objective exactly, and the rule is also consistent. The criterion puzzle then is a representation of the intended criterion performance, and can serve as the final step in the teaching sequence.

BEST COPY AVAILABLE

Puzzle 1, Remedial Level

Domain: Two numbers up to 9 with a "?" in between

Range: The signs >, <, and =

Rule: Type the signs which must replace the "?" to make a true statement.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR MORE THAN
AND = FOR EQUAL TO

2 ? 2 BECOMES =
1 ? 5 BECOMES <
8 ? 4 BECOMES >

Puzzle 2, Remedial Level

Domain: Two numbers up to 99 with a "?" in between

Range: The signs >, <, and =

Rule: Type the sign which must replace the "?" to make a true statement.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR MORE THAN
AND = FOR EQUAL TO

36 ? 8 BECOMES >
4 ? 87 BECOMES <
85 ? 85 BECOMES =

Figure 15. Instructional puzzles for adding fractions and comparing the sum to one

BEST COPY AVAILABLE

Puzzle 3, Entry Level

Domain: A fraction with denominator less than 100 and numerator less than 100 and less than twice the denominator.

Range: The signs "<", "=" and ">".

Rule: Type the sign which must replace the "/" to make a true statement.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR MORE THAN
AND = FOR EQUAL TO

1/16 BECOMES <
22/22 BECOMES =

Puzzle 4, Instructional Level

Domain: Two fractions as in Puzzle 3, with a "+" in between.

Range: Fractions with the same denominator up to 100 and numerators less than twice the denominator and less than 100.

Rule: The numerator of the range element is computed by summing the numerators of the two domain functions. The denominator is the same as the denominator of the fractions.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

FRACTIONS MUST HAVE THE SAME DENOMINATOR

15/11 + 1/11 BECOMES 16/11
19/20 + 17/20 BECOMES 36/20

Figure 15 (Cont'd). Instructional puzzles for adding fractions and comparing the sum to one.

BEST COPY AVAILABLE

Puzzle 5, Criterion Level

Domain: Two fractions as in Puzzle 3, with a "+" in between.

Range: Fractions as in Puzzle 4 followed by one of the signs "<", "=" or ">".

Rule: The fraction is computed as in Puzzle 4, and the sign is that which must replace the "/" of the sum in order to make a true statement.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR MORE THAN
AND = FOR EQUAL TO

FRACTIONS MUST HAVE THE SAME DENOMINATOR

15/11 + 1/11 BECOMES 16/11 >
19/20 + 17/20 BECOMES 36/20 >
2/17 + 5/17 BECOMES 7/17 <

Puzzle 6, Post Criterion Level

Domain: Fractions as in Puzzle 3, with a "+" in between.

Range: The signs "<", "=" and ">".

Rule: The fractions are summed as in Puzzle 4. The range element is the sign needed to replace the "/" of the sum in order to make a true sentence.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR MORE THAN
AND = FOR EQUAL TO

FRACTIONS MUST HAVE THE SAME DENOMINATOR

18/23 + 21/23 BECOMES >
1/4 + 2/4 BECOMES <

Figure 15 (Cont'd). Instructional puzzles for adding fractions and comparing the sum to one

BEST COPY AVAILABLE

Puzzle 7, Post-Criterion Level

Domain: A 1 followed by "?" and a fraction with denominator less than 100 and a numerator less than 100 and less than twice the denominator.

Range: The signs "<," "=" and ">."

Rule: Type "=" if the numerator and denominator are equal, type ">" if the numerator is less than the denominator, and type "<" if the numerator is greater than the denominator.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR GREATER THAN
AND = FOR EQUAL TO

1 ? 64/42 BECOMES <
1 ? 18/22 BECOMES >

Puzzle 8, Post-Criterion Level

Domain: A fraction and the numeral "1" with a "?" between. The "1" may be on the right or left of the fraction.

Range: The signs "<," "=" and ">."

Rule: For domain elements with "1" on the left, follow rule of Puzzle 7; with "1" on the right, ignore the 1 and type the sign needed to replace the "/" of the fraction to make a true sentence.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR GREATER THAN
AND = FOR EQUAL TO

13/13 ? 1 BECOMES =
1 ? 8/12 BECOMES >

Figure 15 (Cont'd). Instructional puzzles for adding fractions and comparing the sum to one.

BEST COPY AVAILABLE

Puzzle 9, Post-Criterion Level

Domain: An indicated sum of two fractions as in Puzzle 4, and the numeral "1" separated by "?" The "1" may be on the right or left of the sum.

Range: The signs "<," "=" and ">."

Rule: Add the two fractions as in Puzzle 4, and replace the indicated sum by the actual sum in the domain. Then follow the rule in Puzzle 8.

Description for Student:

YOU ARE LEARNING TO ADD FRACTIONS WITH THE SAME DENOMINATOR AND COMPARE TO 1

USE < FOR LESS THAN
> FOR GREATER THAN
AND = FOR EQUAL TO

FRACTIONS MUST HAVE THE SAME DENOMINATOR

1 ? 35/47 + 14/47 BECOMES <
6/14 + 7/14 ? 1 BECOMES <

Figure 15 (Cont'd) Instructional puzzles for adding fractions and comparing the sum to one

BEST COPY AVAILABLE

Preparing the Instructional Puzzles. The criterion level puzzle is too complex for a student who has never seen the notation or rules for fractions and has never added two fractions. One less complex instructional puzzle (Puzzle 4) requires the student to add two fractions without doing the comparison. Puzzle 4 uses the same domain as the criterion puzzle, but has a simpler range and rule.

Even Puzzle 4 will be complex for a student just learning to add fractions. A preliminary puzzle (Puzzle 1) teaches the student to recognize the symbol for a fraction and to compare a single fraction to one. The domain is fractions less than two with numerators and denominators less than 100. The range is the set of symbols ($<$, $=$, $>$) for less than, equal to, and greater than. The rule requires the student to compare the numerator and denominator of the fraction and supply the appropriate sign. Since the student has already compared numbers without the fraction sign, additional complexity of the rule should be minimal.

The instructional sequence begins with Puzzle 1 and continues to Puzzle 5. In the first step, the student learns to recognize and type a fraction for the domain element, and applies his knowledge of comparing numbers to the numerator and denominator of the fraction. The next step teaches the student to add two fractions and introduces a new symbol to the domain which consists of two fractions separated by a plus ($+$) sign. The criterion puzzle combines the addition and comparison rules into one. The student adds the two fractions of the domain as he did in the preceding step, and then compares the sum to one as he did in the initial teaching puzzle. The normal teaching sequence consists of these three steps.

Preparing the Remedial Sequence. Some students experience difficulty making the first induction from the rule they know for comparing

BEST COPY AVAILABLE

two numbers up to 99 to the rule for comparing a fraction to one. Two remedial puzzles help such students.

The first remedial puzzle (Puzzle 2) requires the student to recognize and recall the rule he knows to compare numbers less than 100. The symbol [ ] marks the position where the correct sign should be substituted. In Puzzle 2, the only new learning concerns the format of the domain and the range; some students take several trials to learn to type the range symbols correctly.

In the rare cases where remedial Puzzle 2 fails to help the student recall his knowledge of comparing numbers, Puzzle 1 usually helps. Puzzle 1 is slightly simpler, since the domain elements are smaller numbers (up to 9). The rule for Puzzle 1 is identical to the rule for Puzzle 2, but the comparison is easier and more familiar, so the student focuses his attention on the domain and range symbols; this prepares the student for more complex rules he will encounter in Puzzles 2 through 5.

Preparing the Post-Criterion Sequence. At the post-criterion end of the teaching sequence, the performance is extended to more difficult, but closely related skills. Puzzles 6 through 9 in Figure 15 extend the criterion performance. In Puzzle 6, the fractions must be summed, but only the comparison to one is written as the range element. Here the student practices the skill he learned in Puzzle 5 after he recalls it and applies it to the new situation.

The rules for comparing fractions to one that the student induces in Puzzles 5, 7, and 8 depend on the fact that the numeral one is assumed to be to the right of the fraction in the comparison. The rule is much simpler than the more general case where the "1" may be either on the left or right. Comparing 2/3 to 1 is the same as comparing 2 to 3, as

BEST COPY AVAILABLE

long as the 1 is on the right. The situation is different if the numeral "1" is written explicitly on the left of the fraction or randomly on the right or left. Puzzle number 7 introduces new complexity to the domain and to the rule. The student must reverse his usual algorithm for comparing a fraction to 1, and must notice the new symbol in the domain. In Puzzle 8, the position of the 1 can be either to the left or right of the fraction. The student must generalize his rule for making the comparison, thus generalizing the skill he has learned. Finally, Puzzle 9 combines the addition of two fractions with the generalized comparison rule. This puzzle is more difficult because it combines two previously learned rules.

Summary of Example. This section explained the nine puzzles in the teaching sequence for a typical objective. The increase of difficulty between each puzzle is small, allowing the student to induce the new rule from previously familiar rules and the examples provided in input and test modes. In this objective, the domain becomes increasingly complex at the same time the rule gets more complex.

D. MATH FUNCTIONS and the School

The MATH FUNCTIONS program was designed to operate on the small computer resource at the Oakleaf Elementary School, a school which has cooperated in developing educational material with the Learning Research and Development Center since 1964. The program is one component of the Individualized Mathematics system which has operated in the Oakleaf school since 1966. This section describes the relevant components of the computer resource and the relevant aspects of the management of instruction under Individualized Mathematics. These aspects are all normal operating procedures at Oakleaf, and required no special modification when MATH FUNCTIONS was integrated with them. Certain special procedures were required to assure that evaluation data were available. These procedures will be described at the end of this section.

BEST COPY AVAILABLE

1. The Small Computer Resource

Oakleaf School has had computer support for instruction, testing, and management procedures since 1968. Originally, the computer was controlled outside of the school, and could not be fully integrated with the demands of the school. In Spring 1972, a small computer resource (a PDP 15 with 32K core storage) was installed in a van outside of Oakleaf, and now provides locally controlled service to the school throughout the day. In spite of the computer's small size, it has an extensive time sharing system (ETSS), designed and implemented by the Learning Research and Development Center (Fitzhugh, 1970). ETSS currently supports 16 stations at which students work independently.

The fact that MATH FUNCTIONS is a computer program rather than a text causes no special problems at Oakleaf School. The computer resource is a part of the school environment, not a special research tool. The computer stations are located in the regular instructional areas, and every student knows how to use the stations. Whenever a program is relevant to a student's work, whether he is learning new material, testing himself, or recording his progress in the curriculum, he goes to any available station and runs the appropriate program. Specially prepared flowcharts in the vicinity of the computer stations tell the student all he needs to know to run the program he has selected. A flowchart was prepared for MATH FUNCTIONS, and it became part of the computer resource.

The Management of the Individualized Mathematics System. A previous section described the features of the Individualized Mathematics system which make it particularly suitable as a base for MATH FUNCTIONS. These included the individualized, self-paced instruction, behavioral specification of content, guaranteed mastery learning of all skills, and the clear statement of the prerequisite skills for each new skill. This section describes the school based procedures which allow every

BEST COPY AVAILABLE

student to proceed at his own pace and guarantee mastery learning of all skills. It also describes how the new instructional component, MATH FUNCTIONS, was added to the system.

The Individualized Mathematics system is based on approximately 400 instructional objectives. The procedures for self-pacing, instruction, and guaranteed mastery operate identically for each objective. The rest of this discussion focuses on a single objective, and describes all procedures in terms of that objective.

When a student has mastered all the prerequisites for an objective, he takes a competency test to determine if he can also perform the objective. Some students do generalize skills without instruction. If he passes the test, no instruction is needed, and he moves on to the next objective. If he fails, demonstrating his need for instruction, he and his teacher examine all the available instructional materials for the objective. These might include a programmed booklet, a practice program, manipulative materials and a MATH FUNCTIONS teaching sequence. The student and teacher confer and write down an instructional prescription. The student proceeds to do the prescription, usually taking two or three math periods to do so. When the prescription is complete, the student takes a new competency test. If he has learned, he goes on to the next objective. If not, he and his teacher write a new prescription.

Several features of these procedures are important: The student does not need to go over what he already knows simply because others in his class do not know it. The student skips objectives he knows. The prescription is written specifically for the student in consultation with him. He, therefore, does as much work as he needs, and can select the manner in which he works. Finally, the student must demonstrate competency on a performance criterion test before advancing to a new objective, thus assuring that he knows the material.

BEST COPY AVAILABLE

Integrating a new teaching sequence from MATH FUNCTIONS with the established management procedures only required that the teachers and students know that the sequence is available; it is then considered as an option whenever a prescription is required for the relevant objective. Thus, adding a teaching sequence for an objective is a simple matter, and can be handled by routine procedures

MATH FUNCTIONS teaching sequences terminate with a puzzle equivalent to the criterion performance for the objective. Mastery of a teaching sequence encompasses both the instruction and the competency testing aspects of the Individualized Mathematics system. Once the MATH FUNCTIONS program declares that the student has mastered an objective, he can go immediately to the next objective. Use of the computer program to both instruct and test saves considerable time over normal Individualized Mathematics procedures.

Special Procedures for Formative Evaluation. When MATH FUNCTIONS was introduced to Oakleaf in Spring 1972, it was expected to undergo rapid changes as students reacted to the unusual style of the instruction. The teaching sequences could also require modification if students did not find the steps between puzzles small and manageable.

Extensive feedback to the researchers was built into the program to help make instructional design decisions. In addition, adult supervision and observation procedures were used. Finally, all students completing MATH FUNCTIONS were required to take the normal, paper and pencil, curriculum embedded test (CE 1) as a transfer test after completing an objective on MATH FUNCTIONS. This section describes the special procedures introduced to help modify the program as students learned from it.

Procedures internal to the program recorded every student interaction, including the precise inputs he made, the test items he responded

BEST COPY AVAILABLE

to and his answers. The data collected were sufficient to reproduce the entire interaction on paper so that we could examine it in detail in the laboratory. These interactions were printed every day, and two of us examined them and made changes to teaching sequences we deemed necessary.

Additional procedures recorded summaries of the interaction with each student, including the sequence of puzzles he saw and the result of his work on each puzzle. This told us if the students were passing the instructional puzzles without recourse to the remedial puzzles, whether the remedial puzzles helped the students, and whether post-criterion puzzles were passed. This information often led to revision of teaching sequences.

In Spring 1972, a school aide watched every student using the program and reported the observations to us. When students found difficulty with the program, the aide helped and reported her interaction to us. On most days an observer from the research staff also watched the interaction. Together, the report of the school aide and the research staff observer provided further information to help us clarify the teaching sequences.

Finally, the curriculum embedded tests were administered as a transfer test from the instruction. Implicit, guided discovery teaching, such as the MATH FUNCTIONS program provided, should encourage transfer to paper and pencil form of the same skill and of closely related skills. We expected a high success rate on the curriculum embedded tests, and would modify teaching sequences whenever students failed to transfer their learning to the CET. The specific errors students made on the CET's suggested additional puzzles for the teaching sequences.

In Spring 1972, when MATH FUNCTIONS first entered the school, only 22 objectives were used. Because there were so few objectives

BEST COPY AVAILABLE

available, and because we wanted to gain experience with the program quickly, we encouraged teachers and students to select MATH FUNCTIONS as their prescription whenever the objective was available. Teachers were highly cooperative in this initial tryout, notwithstanding the unusual procedures associated with it.

In Summer 1972, 100 additional teaching sequences were prepared, and the MATH FUNCTIONS program itself had several improvements made in the way it judged success or failure on a puzzle and on individual test items. In Fall 1972, a full scale tryout began. No school aide was required for supervision, but a research staff observer was frequently at the school. Teaching sequences were modified throughout the year whenever the need became apparent.

During Summer 1973, further modifications of teaching sequences took place, some of the less successful sequences were withdrawn from the school. In Fall 1973, the program was reintroduced at Oakleaf, and now runs without special supervisory procedures of any sort. The next section describes the results attained from using the program during 14 months in the school.

## III. The Results

This section examines the success of MATH FUNCTIONS in teaching mathematics content. It focuses on results within the program defined criterion performance and on the transfer task, the paper and pencil, curriculum embedded test (CET). Some descriptive statistics on amount of usage are also reported.

### A. Outcome Statistics

The basic datum of this section is an outcome on a single objective for a single student, for example, student number 3333 working on the

BEST COPY AVAILABLE

addition of fractions objective mastered the criterion puzzle and passed the CET. The same student will be involved in several data; the same objective will be involved in several data. No attempt is made here to isolate the effects due to individual students or to individual objectives.

One of two outcomes occurs each time a student works on an objective in MATH FUNCTIONS: The student masters the criterion puzzle (mastery); or the student does not master the criterion puzzle (non-mastery). A further refinement of the outcomes occurs for students who take the curriculum-embedded test (CET) as a transfer task: the student passes the CET (mastery-pass); or the student fails the CET (mastery-fail).

The frequency of the various outcomes is reported in Table 3. For the moment, only the overall totals are relevant. Of the total sample of 601 outcomes since Spring 1972, 529 (88.0 percent) were masteries.

TABLE 3

Outcomes on Objectives

| | Non-Mastery | Mastery | Mastery Fail | Mastery Pass |
|---|---|---|---|---|
| Overall | 72 (12.0%) | 529 (88.0%) | 102 (21.1%) | 381 (78.9%) |
| Spring 1972 | 9 (8.3%) | 99 (91.7%) | 22 (25.0%) | 69 (75.0%) |
| Fall 1972 | 15 (14.7%) | 87 (85.3%) | 23 (33.7%) | 59 (66.3%) |
| Winter & Spring 1973 | 42 (13.1%) | 279 (86.9%) | 56 (18.4%) | 197 (81.1%) |
| Fall 1973 | 6 (8.6%) | 64 (91.4%) | 6 (9.7%) | 56 (90.3%) |
| Grades 3 and 4 | 16 (5.4%) | 243 (94.6%) | 14 (6.2%) | 213 (93.8%) |
| Grades 5 and 6 | 47 (20.1%) | 187 (79.9%) | 65 (39.6%) | 99 (60.4%) |

BEST COPY AVAILABLE

Some students (46) who mastered the objective did not take the CET transfer test. Of those who did take the test, 381 of 483 (78.9 percent) passed it, indicating a high transfer rate from the program defined criterion task to the paper and pencil task.

Students using the paper and pencil teaching sequence pass the CET on the first attempt 85 to 95 percent of the time. The slightly lower passing rate for FUNCTIONS (79 vs. 90 percent) is explained by the fact that FUNCTIONS students took the CET in a mode (paper and pencil) different from their instruction (computer).

Since the MATH FUNCTIONS program changed during the period of this evaluation, a partition of the results across time should reveal if changes affected the outcomes. Four natural divisions occur; Spring 1972 with only 20 objectives in a highly supervised setting; Fall 1972 with 120 objectives in a moderately supervised setting; Winter and Spring 1973 with 108 objectives in a lightly supervised setting; and Fall 1973 with 90 objectives in a lightly supervised setting. Outcomes for these periods are reported separately in Table 3. A Chi Square test of association of nonmastery vs. mastery with time is not significant, making the best estimate of the pass rate 88.0 percent, the overall rate. The transfer rate does change over time ($\chi^2 = 15.9$, 3 d.f., $p < .05$).

The only drop in transfer rate occurs between Spring 1972 and Fall 1972 when six times the number of teaching sequences were available, and the amount of program supervision decreased. Notice a parallel drop in the mastery rate at the same time period. The mastery-pass rate in Fall 1973 is significantly greater than all other time periods ($\chi^2 = 5.6$, 1 d.f., $p < .05$), and is best estimated at 90.3 percent. The higher transfer rate results from revision of teaching sequences and selective elimination of objectives with high mastery-fail rates.

BEST COPY AVAILABLE

A second way to partition the data is by grade level. Student outcomes from grades 3 and 4 are contrasted with outcomes in grades 5 and 6 in Table 3. There is a highly significant relationship ($\chi^2 = 21.3$, 1 d.f., $p < .05$) between the respective mastery rates, favoring the younger students with 94.6 percent mastery over the older students with 79.9 percent mastery. A further difference occurs on the transfer task. Students in grades 3 and 4 have much higher transfer rates than students in grades 5 and 6 ($\chi^2 = 66.1$, 1 d.f., $p < .05$). For the younger grades 93.8 percent of the students transfer; for the older grades only 60.4 percent transfer.

Many factors are confounded with grade level, so these results must be interpreted cautiously. Several of the factors are described below. First, the younger students cover simpler rules with less complex domains. These simpler rules may be easier to teach using MATH FUNCTIONS than the more complex rules higher in the Individualized Mathematics curriculum. Second, the younger students needed to have perfect CET's before advancing to a new objective, in contrast to 85 percent for students in the older grades. While this fact makes the transfer of 93.8 percent even more remarkable, the importance may lie in effect on prerequisite skills. Mastery of prerequisites may be better assured with the higher criterion for advancement. Third, the younger students have used special curricula designed by Learning Research and Development Center since they began school; the older students have used fewer of the special curricula. The better performance on MATH FUNCTIONS may reflect this special training (Rosner, 1973).

BEST COPY AVAILABLE

B. Usage Statistics

This section examines two measures of program usage: the number of objectives learned for each student, and the mean time to complete an objective.

The number of objectives seen by each student is important for two reasons: first, it gives an indication of how interested students are in learning by solving problems and second, it measures how much exposure each student has had to help him learn problem solving skills. One hundred seventy-six students were eligible to work on MATH FUNCTIONS during the 1972-73 school year. One hundred forty-nine or 84.7 percent used the program for at least one objective. The mean number of objectives seen was 3.05, with some students seeing as many as 13 objectives. This amount of usage indicates that students do return several times to the program. A frequency distribution of number of objectives seen by number of students is contained in Figure 16.

It is difficult to determine what fraction of available objectives students chose to work under MATH FUNCTIONS, since students vary enormously in how much they study and precisely what objectives they study. On the average, we estimate that students worked 60 objectives in the school year 1972-73, and pretested out of 40 percent of them, taking instruction in 36 objectives. One-fourth of those, or nine objectives, was available under MATH FUNCTIONS. On the average then, students took one-third of the objectives which were available to them under MATH FUNCTIONS. These figures are rough estimates, but they do give an idea of usage frequency.

The time to complete objectives is important since teachers and students expect to complete objectives in one or two class periods. If the MATH FUNCTIONS program took much longer to teach, it would not

BEST COPY AVAILABLE

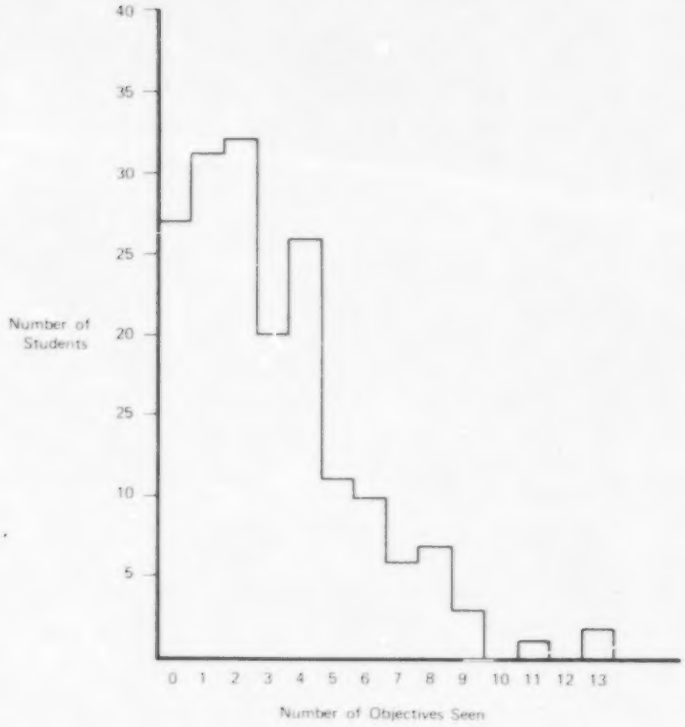


Figure 16. Frequency of students seeing each number of objectives in a year.

BEST COPY AVAILABLE

be acceptable. The actual data indicate that speed is satisfactory, especially during Fall 1973 when a revised and faster version of MATH FUNCTIONS was introduced. The mean time to complete an objective with the old version of the program was 56 minutes, or just over one class period. With the revised program, the average time is 30.0 minutes, less than one class period. We estimate that the average time using the standard teaching sequence is approximately 60 minutes. Even though the MATH FUNCTIONS program teaches problem solving in addition to mathematics content, it takes no longer to teach the content.

## IV. Summary and Implications

The results reported in the last section demonstrate that mathematics content can be taught in an environment which encourages problem solving. This section summarizes the procedure and the conclusions and points out directions for further work.

### A. Summary

The FUNCTIONS program was designed as a problem solving environment. Features were included to teach skills such as stating the problem, gathering data, organizing data, using feedback, subdividing the problem, integrating subsolutions, and knowing when you are finished. Important features include the organized display, the two modes (input and test) of interaction, and the lack of compulsion in the program.

Since mathematics contains many examples of functions, much of elementary school mathematics can be taught using the FUNCTIONS program. Special features of the Individualized Mathematics system such as self-paced learning, clear behavioral objectives and prerequisites, and mastery criteria make it especially adaptable to computerized, individual instruction.

BEST COPY AVAILABLE

One hundred objectives of the Individualized Mathematics system were analyzed, and teaching sequences were prepared for use with the MATH FUNCTIONS program. Each teaching sequence consists of an ordered set of puzzles each of which is slightly more complex than its predecessor.

The MATH FUNCTIONS program was easily integrated into normal operating procedures at the Oakleaf Public School. Oakleaf already had a computer system which the students used regularly and the management procedures of the Individualized Mathematics system easily included new options for teaching objectives.

In over one year of continuous use, students have mastered 88.0 percent of the objectives they study, and have transferred their knowledge to a paper and pencil test 78.9 percent of the time. Improvements in the program and teaching sequence have increased the transfer rate to 90.1 percent during the current semester.

Most students (84 percent) have tried the program and have taken an average of 3.05 objectives. This indicates that many of the students learn from the program. The time to learn is no longer than more traditional methods require.

B. Implications

While the overall success of the program in teaching mathematics content has been demonstrated, much work remains to be done. In this section, some additional lines of investigation are developed. These include: determining if student aptitudes interact with the treatment; developing a model of ideal problem solving in the program and demonstrating that students approach the ideal with experience; clarifying the methodology for creating teaching sequences; and, finally, developing

BEST COPY AVAILABLE

additional programs of a similar nature to broaden the scope of application. The following paragraphs develop these ideas.

It seems likely that students with different entering abilities perform differently in MATH FUNCTIONS, both with respect to outcome and transfer and with respect to problem solving. Once such effects are identified, an effort will be made to determine how the specific aptitudes and abilities affect performance. Possibly, special training procedures can be developed to overcome certain handicaps.

The FUNCTIONS program was designed to encourage certain problem solving skills by its structure. The ideal student will respond to the environment to make his problem solving more efficient. One extension of this work is to develop a model of ideal student behavior and to identify students who exhibit that behavior. The program should shape student behavior toward the ideal; detailed protocol analysis of students at work will be undertaken to identify the specific events which produce behavior changes in the desired direction. When such events are found, contingencies can be built into the program to make especially fruitful events more likely to occur. Analysis such as this will lead to a more powerful procedure for teaching problem solving.

While the methodology for doing the task analysis and creating a teaching sequence follows accepted instructional principles, it remains an art rather than a science. One effort that seems promising is an attempt to further clarify how functions are made slightly more complex in teaching sequences. Formative evaluation work has created a collection of teaching sequences which work and a smaller collection of sequences which are too complex. Analysis of these collections should clarify the differences, and make it possible to state rules more precisely.

BEST COPY AVAILABLE

The success of this one effort at combining the teaching of problem solving with the teaching of mathematics content suggests that other similar efforts should be undertaken. What kinds of programs can be adapted to these uses? What other fundamental ideas occuring in mathematics can be exploited as functions have been in MATH FUNCTIONS? Additional work in this area will attempt to answer these and other important questions.

BEST COPY AVAILABLE

References

Block, K. K., Carlson, M., Fitzhugh, R. J., Hsu, T., Jacobson, E., Puente, G., Roman, R. A., Rosner, J., Simon, D. P., Glaser, R., and Cooley, W. W. A computer resource for the elementary school: Progress Report 1971-1972. Pittsburgh: University of Pittsburgh, Learning Research and Development Center, 1973.

Cooley, W. W., and Glaser, R. The computer and individualized instruction. Science, 1969, 166, 574-582.

Fitzhugh, R. Learning Research and Development Center experimental time-sharing system reference manual, 3 volumes. Pittsburgh: University of Pittsburgh, Learning Research and Development Center, 1970.

Lindvall, C. M. A teacher's manual for individualized mathematics. Pittsburgh: University of Pittsburgh, Learning Research and Development Center, 1973.

Lindvall, C. M., and Bolvin, J. O. Individually Prescribed Instruction: The Oakleaf project. Pittsburgh: University of Pittsburgh, Learning Research and Development Center, 1966. Working Paper 8.

Lindvall, C. M., & Cox, R. C. Evaluation as a tool in curriculum development: The IPI evaluation program. Chicago: Rand McNally, 1970.

Rosner, J. Changes in first grade achievement and the predictive validity of I.Q. scores, as a function of an adaptive instructional environment. Pittsburgh: University of Pittsburgh, Learning Research and Development Center, 1973. Publication 1973/5.